Лабораторна робота №4

КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ. ОРГАНІЗАЦІЯ УМОВНИХ ПЕРЕХОДІВ

Мета роботи: вивчення організації адресного простору пам'яті програм мікроконтролера AT90S2313, програмних засобів керування ходом виконання програми, отримання навичків програмування циклічних алгоритмів в кодах мікроконтролера.

Теоретичні відомості

1. Програмна модель AT90S2313

Так само, як і в інших мікроконтролерів, що вбудовуються, система команд AVR включає команди арифметичних і логічних операцій, команди передачі даних, команди, що керують послідовністю виконання програми і команди операцій з бітами.

Маючи 16-розрядну комірку пам'яті програм, AVR відрізняються багатством своєї системи команд у порівнянні з іншими RiSC-мікроконтролерами.

Для зручності написання й аналізу програм всім операціям із системи команд крім двійкового коду зіставлені мнемокоди Ассемблера (символічні позначення операцій), що використовуються при створенні вихідного тексту програми. Спеціальні програми-транслятори переводять потім символічні позначення в двійкові коди.

Спеціальна директива ассемблера

.device <типAVR>

забезпечує контроль відповідності команд, використовуваних у тексті програми, типу зазначеного процесора.

При переході від молодших до старших моделей AVR існує сумісність у змісті системи команд, однак необхідно пам'ятати, що адреси векторів переривання тих самих периферійних вузлів у різних типів AVR різні, що вимагає внесення відповідних змін у програму при її переносі на інший тип AVR.

На рис. 4.1 зображена програмна модель AVR-мікроконтролерів, що являє собою діаграму програмно доступних ресурсів AVR. Центральним блоком на цій діаграмі є реєстровий файл на 32 оперативних регістра (R0-R31), безпосередньо доступних ALU. Старші регістри об'єднані парами й утворять три 16-розрядних регістри, призначених для непрямої адресації комірок пам'яті .

Всі арифметичні і логічні операції, а також частина операцій роботи з бітами виконуються в ALU тільки над вмістом оперативних регістрів. Варто звернути увагу, що команди (SUBI, SBCI, ANDI, ORI, SBR, CBR) як другий операнд використовують константу, як перший операнд використовуються тільки регістри з другої половини реєстрового файлу (R16-R31). Команди 16-

розрядного додавання з константою ADI і віднімання константи SBI у якості першого операнда використовують тільки регістри R24, R26, R28, R30. Під час виконання арифметичних і логічних операцій чи операцій роботи з бітами ALU формує ті чи інші (див. таблицю 1) ознаки результату операції, тобто чи встановлює скидає біти в регістрі стану SREG (Status Register).

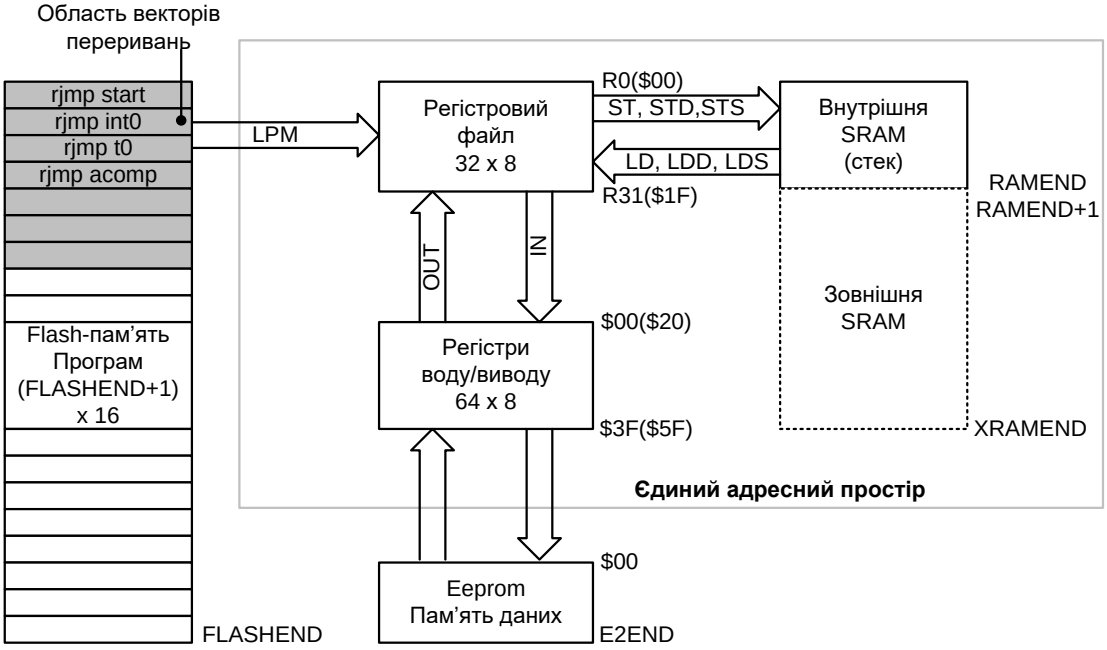

Рисунок 4.1 – Програмна модель AVR-мікроконтролерів

2. Регістр статусу - SREG

Регістр статусу - SREG - розміщений у просторі I/O за адресою $3F ($5F) і його біти визначаються як показано в табл. 4.1.

Bit 7 - I: Global Interrupt Enable - Дозвіл глобального переривання. Біт дозволу глобального переривання для дозволу переривання повинний бути встановлений у стан 1. Керування дозволом конкретного переривання виконується регістрами маски переривання GIMSK і TIMSK. Якщо біт глобального переривання очищений (у стані 0), то жодне з дозволів конкретних переривань, встановлених у регістрах GIMSK і TIMSK, не діє. Біт I апаратно очищається після переривання і встановлюється для наступного дозволу глобального переривання командою RETI.

Bit 6 - T: Bit Copy Storage - Біт збереження копії. Команди копіювання біта BLD (Bit Loa) і BST (Bit STore) використовують біт T як біт джерело і біт призначення при операціях з бітами. Командою BST біт регістра реєстрового файлу копіюється в біт T, командою BLD біт T копіюється в регістр реєстрового файлу.

Bit 5 - H: Half Carry Flag - Прапор напівпереносу. Прапор напівпереносу вказує на напівперенос у ряді арифметичних операцій

Bit 4 - S: Sign Bit, S = N V - Біт знака. Біт S завжди знаходиться в стані, обумовленому логічним що виключає ЧИ (exclusive OR) між прапором негативного значення N і доповненням до двох прапора переповнення V.

Bit 3 - V: Two's Complement Overflow Flag . Доповнення до двох прапора переповнення. Доповнення до двох прапора V підтримує арифметику доповнення до двох.

Bit 2 - N: Negative Flag - Прапор негативного значення. Прапор негативного значення N вказує на негативний результат ряду арифметичних і логічних операцій.

Bit 1 - Z: Zero Flag -Прапор нульового значення. Прапор нульового значення Z вказує на нульовий результат ряду арифметичних і логічних операцій.

Bit 0 - C: Carry Flag -Прапор переносу. Ознаки результату операції можуть бути використані в програмі для виконання подальших арифметично-логічних операцій чи команд умовних переходів.

Таблиця 4.1 – Регістр статусу SREG

Біти	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
$3F ($5F)	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
Читання/Запис	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Для збереження оперативних даних програміст, крім реєстрового файлу, може використовувати внутрішню і зовнішню (якщо вони маються) блоки SRAM (рис. 4.1).

Операції обміну з внутрішньою оперативною пам'яттю AVR-мікроконтролер виконує за два машинних цикли. Доступ до зовнішнього SRAM вимагає одного додаткового циклу на кожен байт у порівнянні з внутрішньою пам'яттю. Крім того, установкою біта SRW у регістрі вводу/виводу MCUSR можна програмно збільшити час обміну з зовнішньої SRAM ще на один додатковий машинний цикл очікування.

Виконувати арифметично-логічні операції й операції зрушення безпосередньо над змістом комірок пам'яті не можна. Не можна також записати чи константу очистити вміст комірки пам'яті. Система команд AVR дозволяє лише виконувати операції обміну даними між осередками SRAM і оперативними регістрами. Достоїнством системи команд можна вважати різноманітні режими адресації комірок пам'яті.

Усі регістри введення/виведення можуть зчитуватися і записуватися через оперативні регістри за допомогою команд IN, OUT (див. групу команд передачі даних). Регістри вводу/виводу, що мають адреси в діапазоні $00 - $1F (знак $ указує на шістнадцяткову систему числення), мають можливість побітової адресації. Безпосередня установка і скидання окремих розрядів цих регістрів виконується командами SBI і CBI (див. групу команд роботи з бітами). Для ознак результату операції, що є бітами регістра вводу/виводу SREG, мається цілий набір команд установки і скидання. Команди умовних переходів у якості своїх операндів можуть мати як біти-ознаки результату

операції, так і окремі розряди побітно адресованих регістрів введення/виведення.

На рис. 4.1 показаний розподіл адрес у єдиному адресному просторі. Молодші 32 адреси ($0 - $1F) відповідають оперативним регістрам. Наступні 64 адреси ($20 - $5F) зарезервовані для регістрів вводу/виводу. Внутрішня SRAM у всіх AVR починається з адреси $60.

Таким чином, регістри введення/виведення мають подвійну нумерацію. Якщо використовуються команди IN, OUT, SBI, CBI, SBI, SBI, то варто використовувати нумерацію регістрів введення/виведення, що починається з нуля (назвемо її основний). Якщо ж до регістрів введення/виведення доступ здійснюється як до комірок пам'яті, то необхідно використовувати нумерацію єдиного адресного простору оперативної пам'яті даних AVR. Очевидно, що адреса в єдиному адресному просторі пам'яті даних виходить шляхом додатка числа $20 до основної адреси регістра введення/висновку.

Крім оперативної пам'яті програмно доступними ресурсами мікроконтролера є енергонезалежні, електрично програмувальні FLASH і EEPROM блоки пам'яті, що мають окремі адресні простори.

Тому що всі команди AVR являють собою 16-розрядні слова, FLASH-пам'ять організована як послідовність 16-розрядних осередків і має ємність від 512 слів до 64K слів у залежності від типу кристала.

В FLASH-пам'ять, крім програми, можуть бути записані постійні дані, що не змінюються під час функціонування мікропроцесорної системи. Це різні константи, таблиці знакогенераторів, таблиці лінеаризації датчиків і т.п. Дані з FLASH пам'яті можуть бути програмним образом зчитані в реєстровий файл за допомогою команд LPM, ELPM (див. групу команд передачі даних).

Молодші адреси пам'яті програм мають спеціальне призначення. Адреса $0000 є адресою, з якого починає виконуватися програма після скидання процесора. Починаючи з наступного адреси $0001, комірки пам'яті програм утворять область векторів переривання. У цій області для кожного можливого джерела переривання відведена своя адреса, по якому (у випадку використання даного переривання) розміщають команду відносного переходу RJMP на підпрограму обробки переривання (рис. 4.1). Варто пам'ятати, що адреси векторів переривання тих самих апаратних вузлів для різних типів AVR можуть мати різне значення. Тому для забезпечення сумісності програмного забезпечення зручно, так само як і у випадку з регістрами введення/виведення, використовувати символічні імена адрес векторів переривання, що визначені у відповідному inc-файле.

EEPROM блок пам'яті даних AVR, що електрично стирається, призначений для збереження енергонезалежних даних, що можуть змінюватися безпосередньо на об'єкті. Це калібровані коефіцієнти, різні уставки, конфігураційні параметри системи і т.п. EEPROM-пам'ять даних може бути програмним шляхом як лічена, так і записана. Однак спеціальних команд звертання до EEPROM-пам'яті немає. Читання і запис комірок

EEPROM виконується через регістри вводу/виводу EEAR (регістр адреси), EEDR (регістр даних) і EECR (регістр керування).

3. Команди передачі керування

Групу команд передачі керування утворюють команди безумовного переходу, умовного переходу, команди виклику підпрограми і команди повернення з підпрограми. Характеристики команд приведені в таблиці 4.2.

Таблиця 4.2 - Інструкції розгалуження

Мнем-ніка	**Операнди**	**Опис**	**Операція**		**Цикли**
RJMP	k	Відносний перехід	PC = PC + k +1	None	2
IJMP	-	Непрямий перехід на (Z)	PC = Z	None	2
EIJMP	-	Розширений непрямий перехід на (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Перехід	PC = k	None	3
RCALL	k	Відносний виклик підпрограми	STACK = PC+1, PC = PC + k + 1	None	3/4 *
ICALL	-	Непрямий виклик (Z)	STACK = PC+1, PC = Z	None	3/4 *
RET	-	Повернення з підпрограми	PC = STACK	None	4/5 *
RETI	-	Повернення з переривання	PC = STACK	I	4/5 *
CPSE	Rd,Rr	Порівняти, пропустити якщо рівні	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Порівняти	Rd -Rr	Z,C, N,V, H,S	1
CPC	Rd,Rr	Порівняти з переносом	Rd - Rr - C	Z,C, N,V, H,S	1
CPI	Rd,K8	Порівняти з константою	Rd - K	Z,C, N,V, H,S	1
SBRC	Rr,b	Пропустити якщо біт у регістрі очищений	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3
SBRS	Rr,b	Пропустити якщо біт у регістрі встановлений	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Пропустити якщо біт у порту очищений	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Пропустити якщо біт у порту встановлений	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3

BRBC	s,k	Перейти якщо прапор у SREG очищений	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Перейти якщо прапор у SREG установлений	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	Перейти якщо дорівнює	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Перейти якщо не дорівнює	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Перейти якщо перенос установлений	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Перейти якщо перенос очищений	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Перейти якщо дорівнює чи більше	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Перейти якщо менше	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Перейти якщо мінус	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Перейти якщо плюс	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Перейти якщо більше чи дорівнює (зі знаком)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Перейти якщо менше (зі знаком)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Перейти якщо прапор внутрішнього переносу встановлений	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Перейти якщо прапор внутрішнього переносу очищений	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Перейти якщо прапор T встановлений	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Перейти якщо прапор T очищений	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Перейти якщо прапор переповнення встановлений	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Перейти якщо прапор переповнення очищений	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Перейти якщо переривання дозволені	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Перейти якщо переривання заборонені	if(I==0) PC = PC + k + 1	None	1/2

*Для операцій доступу до даних кількість циклів зазначена за умови доступу до внутрішньої пам'яті даних, і не коректно при роботі з зовнішнім ОЗП. Для інструкцій ICALL, EICALL, RCALL, RET і RETI, необхідно додати три цикли плюс по два циклу для кожного чекання в контролерах з PC меншим 16 біт (128KB пам'яті програм). Для пристроїв з пам'яттю програм понад 128KB , додайте п'ять циклів плюс по трьох циклу на кожне чекання.

Rd: Результуючий (і вихідний) регістр у реєстровому файлі;

Rr: Вихідний регістр у реєстровому файлі;

b: Константа (3 біти), може бути константне вираження;

s: Константа (3 біти), може бути константне вираження;

P: Константа (5-6 біт), може бути константне вираження;

K6: Константа (6 біт), може бути константне вираження;

K8: Константа (8 біт), може бути константне вираження;

k: Константа (розмір залежить від інструкції), може бути константне вираження;

q: Константа (6 біт), може бути константне вираження;

Rdl: R24, R26, R28, R30. Для інструкцій ADIW і SBIW;

X,Y,Z: Регістри непрямої адресації (X=R27:R26, Y=R29:R28, Z=R31:R30).

Завдання до лабораторної роботи

У пам’яті програм, починаючи з комірки ADR2, є таблиця кодів довжиною N. Записати в кодах AT90S2313 програму, яка виконує пересилання даного масиву в SRAM, починаючи з адреси ADR3. Програма повинна починатись з комірки ADR1. Коди задавати довільно

Таблиця 4.3 - Варіанти завдань

Номер	**ADR1**	**ADR2**	**N**	**ADR3**
01	071	C3	E	6F
02	062	C4	F	63
03	053	C2	D	75
04	044	C1	C	86
05	035	C2	B	68
06	026	C3	A	6D
07	017	A4	6	6F
08	078	A5	7	7A
09	069	A6	F	7C
10	05A	A7	E	74
11	04B	A8	5	73
12	03C	B3	6	78
13	02D	BA	7	8C
14	01E	BB	8	6C
15	07E	BC	8	6E
16	06E	6D	9	8A

17	05A	7E	F	6A
18	04D	AF	A	6B
19	08C	BE	B	9B
20	026	CD	C	7B
21	04B	88	D	69
22	03C	63	E	87
23	02D	6A	F	85
24	01E	6B	8	83
25	07F	6C	8	AD
26	06E	62	9	A5
27	05A	7E	C	AE

Зміст звіту

Короткий зміст статусу SREG.
Таблиця команд передачі керування.
Завдання до лабораторної роботи.
Текст програми з поясненнями.

Приклад виконання завдання

У пам'яті команд з адреси ADR2 = $67 розташовано N = $0C кодів. Наприклад: $FF, $00, $11, $22, $33, $44, $55, $66, $77, $88, $99, $AA. Необхідно переписати їх до пам'яті даних, починаючи з адрес ADR3 = $AE. Програма повинна починатися з адреси ADR1 = $05A.

Текст програми:

```
LDI R16,$0C      ;запис до регістру R16 лічильника чисел N
CLR R27          ;скидання змісту верхньої частки регістрової пари  X
LDI R26,$67      ;запис до регістрової  пари  X адреси  ADR2
CLR R29          ;скидання змісту верхньої частки регістрової пари  Y
LDI R28,$AE      ;запис до регістрової  пари Y адреси  ADR3
L1:
LD R1, X         ;завантажити до R1 числа, адреса якого знаходиться в X
MOV R9,R1        ;пересилка змісту R1 в R9
ST Y,R9          ;запис числа з R9 по адресі, яка знаходиться вY (R29R28)
DEC R16          ;зменшити зміст лічильника на 1
INC R26,1        ;збільшити адресу X на 1
INC R28,1        ;збільшити адресу Y на 1
CPI R16,0        ;порівняти зміст R16 з 0
BRNE L1          ;якщо R16 ≠ 0 то перейти по L1
NOP
```

Зміст регістрів після виконання програми

Registers

R0 = 0x00	R15 = 0x00	R30 = 0x00
R1 = 0xAA	R16 = 0x00	R31 = 0x00
R2 = 0x00	R17 = 0x00	
R3 = 0x00	R18 = 0x00	
R4 = 0x00	R19 = 0x00	
R5 = 0x00	R20 = 0x00	
R6 = 0x00	R21 = 0x00	
R7 = 0x00	R22 = 0x00	
R8 = 0x00	R23 = 0x00	
R9 = 0xAA	R24 = 0x00	
R10 = 0x00	R25 = 0x00	
R11 = 0x00	R26 = 0x73	
R12 = 0x00	R27 = 0x00	
R13 = 0x00	R28 = 0xBA	
R14 = 0x00	R29 = 0x00	

Зміст SRAM після виконання програми.

Memory:1

Data | TXT | 8 | 16 | 0x000060

Address	Data
000060	00 00 00 00 00 00 00 FF 00 11 22 33 44 55 66 77
000070	88 99 AA 00 00 00 00 00 00 00 00 00 00 00 00 00
000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 00
0000B0	11 22 33 44 55 66 77 88 99 AA 00 00 00 00 00 00
0000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Контрольні запитання

1. Назвіть призначення бітів регістра статусу SREG.
2. Які види адресації використовуються у програмі?
3. Які команди передачі керування використовуються у програмі?
4. Поясніть програмну модель AVR-мікроконтролера.